AD-764 548

NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR PACKAGE LLRANDOM

Gerard P. Learmonth, et al

Naval Postgraduate School
Monterey, California

June 1973

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

NPS55LW73061A

NAVAL POSTGRADUATE SCHOOL

Monterey, California

NAVAL POSTGRADUATE SCHOOL

RANDOM NUMBER GENERATOR PACKAGE LLRANDOM

by

G. P. Learmonth

and

P. A. W. Lewis

June 1973

48

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

Naval Postgraduate School Random Number Generator Package LLRANDOM

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report

5. AUTHOR(S) (First name, middle initial, last name)

Peter A. W. Lewis and Gerard P. Learmonth

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1973	~~57~~ 48	6

8a. CONTRACT OR GRANT NO.

NR042-284

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

This report is intended to describe an interim version of a computer program package for random number generation on the IBM System/360. The package, when called by a FORTRAN IV program, will deliver either a single value or an array (of specified size) of single precision uniformly, normally, or exponentially distributed pseudo-random deviates, or a single value or an array of uniformly distributed integers between 1 and $2^{31}-1$. The package also has the ability (optional) to "shuffle" the pseudo-random numbers to obtain "better" statistical properties.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Random number generator						
Pseudo-random numbers						
Normal distribution						
Exponential distribution						
Shuffled random numbers						
Division simulation						
Congruential generator						

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral M. B. Freeman
Superintendent

M. U. Clauser
Provost

ABSTRACT

This report is intended to describe an interim version of a computer program package for random number generation on the IBM System/360. The package, when called by a FORTRAN IV program, will deliver either a single value or an array (of specified size) of single precision uniformly, normally, or exponentially distributed pseudo-random deviates, or a single value or an array of uniformly distributed integers between 1 and $2^{31}-1$. The package also has the ability (optional) to "shuffle" the pseudo-random numbers to obtain "better" statistical properties.

Prepared by:

Peter A. W. Lewis*
Department of Operations Research
and Administrative Sciences

Gerard P. Learmonth
Computer Center

Approved by:

J. R. Borsting, ~~Chairman~~
Department of Operations Research
and Administrative Sciences

Released by:

J. M. Wozencraft
Dean of Research

*Partial support by the Office of Naval Research through Contract NR042-284 is gratefully acknowledged.

NPS55LW73061A

NAVAL POSTGRADUATE SCHOOL

RANDOM NUMBER GENERATOR PACKAGE LLRANDOM

I. Introduction.

Numerous random number generators have been proposed for the System/360. Several of these generators have been incorporated into the subroutine library here at the Computer Center. The adequacy of some of these generators has rested on the results of some rather weak tests for randomness; recent results in the literature have shown many of these generators to be very poor performers. This report will describe an interim version of a package for random number generation which has stood up under intensive statistical testing and is deemed to be very satisfactory for the System/360. (The statistical testing will be reported elsewhere.)

The package, when called by a FORTRAN IV program, will deliver either a single value or an array (of specified size) of single precision, uniformly, normally, or exponentially distributed pseudo-random deviates or a single value or an array of pseudo-random integers uniformly distributed between 1 and $2^{31}-1$. The package also has the ability (optional) to "shuffle" the pseudo-random numbers to obtain "better" statistical properties.

Further refinements will be made to this generator; however, it is now available for use in an interim version under the name LLRANDOM. Future versions will be announced through the W. R. Church Computer Center Newsletter. The changes envisioned will be internal and aimed at increasing speed and efficiency of coding. The actual numbers produced in future versions will remain the same as described here, as will the FORTRAN calling sequence.

Some definitions. By "random number generator" or "pseudo-random number generator" is meant an algorithm by which sequences of numbers

are produced which follow a given probability distribution and possess the appearance of randomness. Without attempting to address the still unresolved philosophical question of what a random sequence is, the underlined words above are the keys to random number generation on a digital computer. The term sequence implies that many random numbers must be produced serially from the algorithm. The user may need only a very few of these numbers, however we generally require that the algorithm be able to produce very many numbers. Distribution implies that we can associate a probability statement with the occurrence of each random number. The distribution is usually taken to be uniform, that is, within a given range the probability of occurrence of a given number is the same as for any other number in a similar range. If the algorithm produces, say, m distinct numbers then the probability of occurrence for any one of them is $1/m$.

Lastly, we speak of the appearance of randomness. As will be shown next, the actual implementation of the algorithm is a recurrence relation where each succeeding number is a function of the preceding number. True randomness would require independence of successive numbers; however, the algorithm generates a deterministic sequence. Algorithms for random number generation do, however, yield sequences which appear to be random, hence the term "pseudo-random numbers." It is this characteristic which is the subject of statistical testing, that is, one asks, "how random does the given sequence appear?"

The uniform random number generator which forms the basis of the package described here is a Lehmer congruential generator. The recurrence relation is given by

$$X_n \equiv A \cdot X_{n-1} + C \pmod{m}. \qquad (1)$$

This generator produces integer random numbers between 1 and m. These integer values may then be transformed into real-valued numbers between 0.0 and 1.0 or into any desired distribution by an appropriate transformation.

II. The Generator.

The recurrence relation given in equation (1) is actually called a "Lehmer mixed congruential generator." The term mixed comes from the fact that it involves a multiplication by a constant, A, plus an addition of a constant, C. The actual implementation used in LLRANDOM is called a "multiplicative," or "pure," congruential generator in that we take $C = 0$, giving

$$X_n \equiv A \cdot X_{n-1} \pmod{m}. \qquad (2)$$

The field of positive integers is, of course, infinite. It is a reality of digital computers that only a finite number of positive integers are expressible. Specifically, we are limited to the word size of the System/360. This word size is 32 bits with one bit reserved for the algebraic sign; hence, an obvious choice for m is 2^{31}. The product $A \cdot X_{n-1}$ is formed by the System/360 in two adjacent registers yielding a result which may be as large as 2^{63}. We must, however, reduce this product to a number less than or equal to 2^{31}. The mod, or modulo, operation accomplishes this. The product $A \cdot X_{n-1}$ is divided by 2^{31} leaving a quotient which is some multiple of 2^{31} and a remainder which is strictly less than 2^{31}. It is this remainder which is the next pseudo-random number X_n in the equation (1).

On first examination it would appear that a full 2^{31} numbers could be generated by the sequence (1). This is not the case, unless A and m are chosen properly. We define a term called the <u>period</u> which is the number of unique random numbers computable for a given choice of A and m. To illustrate the concept, assume we have a six-bit word with one bit for a sign. We then have $m = 2^5 = 32$. Choose $A = 9$ and work through a sequence starting with $X_0 = 1$.

Step n	X_{n-1}	$A \cdot X_{n-1}$	$A \cdot X_{n-1} \pmod{2^5}$
1	1	9	9
2	9	81	17
3	17	153	25
4	25	225	1
5	1	9	9
.	.	.	.
.	.	.	.
.	.	.	.

Note that the modulus of this generator is 32, however we have realized a period of only 4, that is the sequence of 1, 9, 17, 25, 1, 9, . . . repeats after only 4 numbers. Obviously, care must be taken to insure that such occurrences do not happen in a random number generator. Hopefully, the period will also be independent of the starting value, X_0.

A great deal of work has been done on number theoretic considerations for the choice of m so as to yield a maximum period length (see Knuth[3]). To summarize, generators with modulus $m = 2^p$ for any integer, p, can have a maximum period of $m/4$, or, for the System/360, $2^{31}/4 = 2^{29}$; the period may also depend on the starting value. When the modulus m is a prime number, the maximum possible period is $m - 1$.

It so happens that the largest prime less than or equal to 2^{31} is $2^{31} - 1$, which is most fortuitous. Hence, choosing $m = 2^{31} - 1$ we can achieve a maximum period of $m - 1 = 2^{31} - 2$. These results produce only upper bounds on the period length. Recall in the example above, the maximum period possible is $2^5/4 = 2^3 = 8$, but that a period of only 4 was observed. This naturally leads to considerations of the choice of the multiplier, A.

Success in achieving a maximum period lies with the choice of the multiplier. Again, to briefly summarize the pertinent number theory, for a modulus 2^{31} the multiplier A must differ by 3 from the nearest multiple of 8; the starting value, X_0, must be odd; A must be one greater than a multiple of 4; and C must be odd. These conditions only assure a maximum period of m/4, not necessarily good statistical properties. For the random number generator described here (LLRANDOM) we are choosing $C = 0$; hence, this length is not achievable if $m = 2^{31}$. Luckily, the conditions on choosing A for the modulus $m = 2^{31} - 1$ are more easily met and we can achieve the maximum period.

Utilizing some of the nice number theoretic properties of the number $2^{31} - 1$, to achieve a maximum period, A must be a positive primitive root of $2^{31} - 1$ or a power of such a number. This is generally not easy to find; the value of A used in the generator described here is 7^5. The number 7 is a positive primitive root of $2^{31} - 1$ and raising 7 to the fifth power results in the multiplier 16807 which is also a positive primitive root of $2^{31} - 1$ (Lewis, Goodman, and Miller[1]) and satisfies some conditions regarding the statistical performance of the generated sequence. These conditions will not be discussed here.

The generator

$$X_n \equiv 7^5 \cdot X_{n-1} \pmod{2^{31}-1} \qquad (3)$$

is the generator reported in Lewis, Goodman, and Miller[1]. The authors cite the results of very extensive tests on this generator, all of which show that it is very satisfactory.

A. Division simulation. A practical consideration for random number generators is that they be fast, hopefully without requiring excessive memory to achieve speed. In many applications rather large quantities of numbers are needed and the speed of the generator can be crucial.

Nearly all random number generators are coded as subroutine or function subprograms in the assembler or machine language of the computer. The algorithm for implementing (3) is rather simple, involving a multiplication and then a division to effect the modulo operation. On most computers the division operation is rather slow as compared to the multiplication operation. In the past, the multiplier A was chosen so that its binary representation contained many zeroes, thereby speeding the multiplication. Unfortunately, this choice was at the expense of the period length, since such multipliers rarely met the number theoretic conditions for a maximum period. For the LLRANDOM generator (3) described here, the division operation has been replaced by a division simulation involving two shifts and an add instruction. Should a fixed-point (integer) overflow occur, two more additions are required to correct the situation.

The ordinary division on a System/360 Model 67-2 requires 8.49 micro-seconds. Without overflow, the simulation requires only 3.45

micro-seconds. When overflow occurs, the simulation takes an additional 2.32 micro-seconds for a total of 5.77 micro-seconds. These overflows occur quite rarely, on the order of only once in 250,000 iterations.

The division simulation algorithm (again due to Lehmer) is discussed by Payne, Rabung, and Bogyo[2] and works as follows. Define a congruence relationship by

$$X_n' \equiv A \cdot X_{n-1} \pmod{2^{31}}. \tag{4}$$

Performing the modulo operation on the product AX_{n-1} would give

$$AX_{n-1} = q2^{31} + r, \tag{5}$$

where q is some quotient and r is the remainder and is strictly less than 2^{31}. Adding q to both sides of (4) we get

$$X_n = X_n' + q \equiv AX_{n-1} \pmod{2^{31}-1}. \tag{6}$$

This form gives the desired modulus of $2^{31} - 1$, if there is no overflow in the addition of $X_n' + q$. If there is overflow, to correct it we merely add a constant of 1 to get

$$X_n \equiv X_n' + q + 1 \pmod{2^{31}} = A \cdot X_{n-1} \pmod{2^{31}-1}, \tag{7}$$

which is again, the desired result.

This division simulation algorithm is very easily implemented on the System/360 and saves considerable execution time over conventional division.

B. Shuffling. The sequence produced by the generator (3) does appear to consist of independent, uniformly distributed numbers for most purposes.

We realize that the numbers are not actually independent, due to the procedure used to generate them. It has been proposed that a sequence of such numbers be further randomized, or "shuffled," to improve upon the appearance of randomness (see, for instance, Knuth[3]). Serial correlation tests are usually employed to detect lack of independence in a sequence and at least one generator, RANDU, known to perform badly in a three-dimensional serial test was improved by shuffling. These tests will be discussed elsewhere. The various shuffling procedures which have been put forward have had little empirical validation.

The package described here has a built-in shuffling mechanism and it works as follows. A table of 128 random integers is maintained in the package. The starting values in the table represent members of the sequence (3) lagged by one million integers starting with an arbitrary seed. When a new integer is generated by the algorithm, its right-most seven bits are masked-off to form an index into the table ($2^7=128$). The integer in the table indexed by the right-most seven bits is returned to the caller and that table entry is replaced by the integer just generated. In essence, we are taking "chunks" of 128 numbers from the basic sequence and shuffling them before they are used.

This particular shuffling scheme is dependent on the choice of the modulus. For a modulus of 2^{31} the right-most bits of a congruential random number generator are non-random and their use in this scheme would defeat the purpose of shuffling. However, with a modulus of $2^{31} - 1$ and the positive primitive root multiplier $A = 7^5$, the right-most bits are quite random and the desired results are obtained.

C. <u>Uniform (0.0,1.0) random numbers</u>. So far, we have discussed how to generate uniform random integers over the range 1 to $m = 2^{31} - 1$. In most applications, uniform random numbers over the range 0.0 to 1.0 are desired. In theory, the uniform integers, X_i, are divided by m to produce these numbers, as

$$U_i = X_i/m. \tag{8}$$

In actual implementation on the System/360, the integer result is algebraically shifted right seven bits and a normalized floating point exponent is logically OR'ed on to it. The result is a properly normalized floating point random number over the range 0.0 to 1.0, usually referred to as a "real" uniform number.

D. <u>Normally distributed random deviates</u>. The uniformly distributed random numbers described above are not only useful in their own right, but form the basis of transformations into random numbers with other probability distributions. One of the most important of these distributions is the Normal distribution.

There are several methods of approximating a Normal distribution with uniform random numbers. One of the oldest and, unfortunately, most common is the "sum of k uniforms method." The algorithm is based on the fact that the uniform (0.0,1.0) distribution has a mean of 1/2 and a standard deviation of $\sqrt{1/12}$. The algorithm works as follows:

$$X = \frac{\sum_{i=1}^{k} U_i - k/2}{\sqrt{k/12.0}} \tag{9}$$

The random deviate X is approximately normally distributed with mean 0 and variance 1. The approximation is not as good as other methods and it is rather time consuming in that k uniforms must be generated and then summed. It was basically devised to overcome the very time consuming multiply and divide operations in older computers.

A more accurate algorithm is known as the Box-Muller method or Polar method which is actually a rejection method due to von Neumann. The method requires the generation of two uniforms to produce two independent Normals. It is based on the distribution of points inside the unit circle. The method is more accurate than the "sum of k uniforms method" (in fact, theoretically perfect). However, it does require two square roots and two natural logarithm operations which are generally rather time consuming.

The algorithm used in the package described here is based on a method developed by Marsaglia and is known as the "rectangle-wedge-tail" method. This algorithm is by far the fastest algorithm available for generating normally distributed random numbers, although it requires more memory than the Polar method.

The second volume of Knuth's "The Art of Computer Programming"[3] gives a complete and detailed description of the algorithm. Briefly, the positive half of the Normal density curve is discretized into 37 rectangles, wedges, and a tail as in Figure 1. All of the rectangles are uniformally distributed densities. The wedges are approximated by "nearly linear densities." Finally, the tail distribution is computed by a modification to the Polar method. The normal density, f(x), is then given by the composite function.

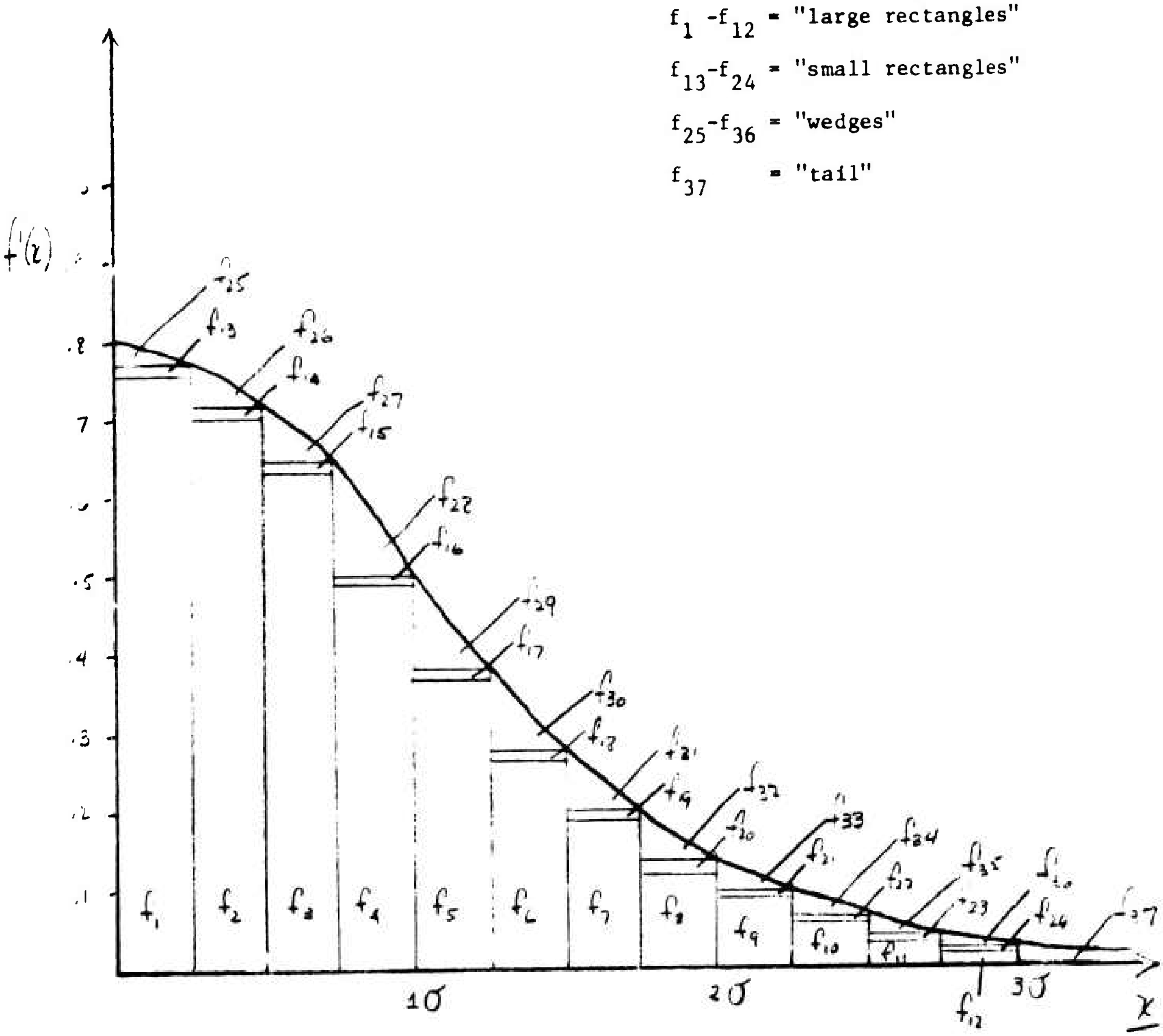

FIGURE 1

RECTANGLE-WEDGE-TAIL METHOD OF APPROXIMATING THE NORMAL DENSITY

$$f(x) = p_1 f_1(x) + p_2 f_2(x) + \ldots + p_{37} f_{37}(x), \tag{10}$$

where

$$\sum_{i=1}^{37} p_i = 1,$$

the densities f_1 to f_{24} are the rectangles; f_{25} to f_{36} are the wedges; and f_{37} is the tail. The first twelve uniformly distributed rectangles are used 88% of the time. This makes for an extremely fast algorithm for the majority of deviates. When the tail is sampled, the deviate is generated by a modified Polar method and still quite satisfactory.

This generator for Normal deviates, like nearly all others, produces deviates with zero mean and unit variance. To change the scale and shape to any mean, μ, and the standard deviation, σ, we apply the linear transformation

$$Z = \mu + \sigma X \tag{11}$$

where z now has the desired shape and scale parameters.

E. Exponential distribution. Another probability distribution of major interest in simulations is the exponential. The cumulative distribution function and probability density function for the exponential are respectively

$$F(x) = 1 - e^{-\lambda x}, \tag{12}$$

$$f(x) = \lambda e^{-\lambda x}. \tag{13}$$

The expected value of the exponential distribution is:

$$E[X] = 1/\lambda. \tag{14}$$

The problem of generating exponential deviates reduces to one of generating "unit" exponentials, i.e. those with $\lambda = 1$, and then multiplying the result by whichever λ is necessary to give the desired distribution.

One of the most common methods of generating numbers from distributions other than the uniform is to use the <u>inverse transformation technique</u> (see Gaver and Thompson[4]). This can be described graphically, as in Figure 2, with a plot of the distribution function

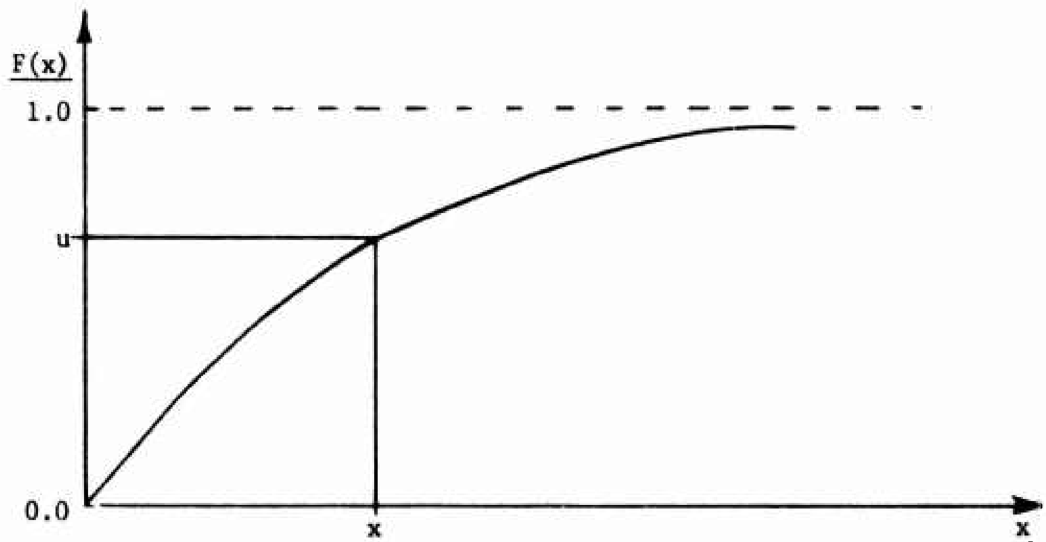

FIGURE 2.

CUMULATIVE DISTRIBUTION FUNCTION OF THE EXPONENTIAL DISTRIBUTION

The range of the abscissa, X, is infinite in extent. However, the range of the ordinate, F(x), is (0.0,1.0), the range of uniform (0,1) random variables. The inverse transformation technique is to generate a uniform random number, say U, and use this as the ordinate. The exponential deviate, say X, is the abscissa point corresponding to the intersection of the ordinate and the curve.

Mathematically, this technique is expressed as

$$u = F(x) = 1 - e^{-x}, \qquad \lambda = 1, \tag{15}$$

$$x = F^{-1}(u),$$

where u is the uniform random number. This inverse transformation is rather easily implemented for exponentially distributed random variables via natural logarithms since we get

$$F^{-1}(U) = -\ell n(1-U),$$

or by the symmetry of the uniform distribution

$$F^{-1}(U) = -\ell n(U).$$

Perhaps the most common implementation of exponential deviate generators is this natural logarithm transformation. It is mathematically appealing as well as trivial to program, given the usual FORTRAN subroutines.

The exponential deviate generator in the LLRANDOM package is based on Marsaglia's method of dividing the probability density into a series of rectangles, wedges, and a tail. Although more complicated to program and larger in size, this method is approximately 40% faster than the logarithmic transformation.

For a survey of the generation of normal and exponentially distributed variables see Ahrens and Dieter[5].

III. HOW TO USE THE PACKAGE.

The random number package described here is intended solely for use on the IBM System/360 or System/370 computers. The package consists of one Assembler F control section (CSECT) with nine entry points and two FORTRAN IV function subprograms. The names of the entry points and their functions are summarized in Table 1.

The subroutine entry point OVFLOW has no calling arguments and should be called once and only once at the beginning of the user's main FORTRAN program. The function subprograms RNORTH and REXPTH are called by the Assembler routine as needed and should not be called by the user. The eight additional entry points are the names of the actual routines to generate the random numbers. There are four types of random numbers which can be generated:

(1) uniformly distributed integers on the range 1 to $2^{31} - 1$;

(2) uniformly distributed single precision floating point numbers between 0.0 and 1.0;

(3) single precision floating point normal deviates with mean zero and variance 1; and

(4) single precision floating point exponential deviates with mean 1.

There is a separate entry point for each of the four types if shuffling of the sequence is desired.

For all eight entry points, the FORTRAN calling sequence is the same, namely:

CALL (entry point) (IX, A, N)

where

(entry point) refers to the routine desired,

viz. INT,SINT,RANDOM,SRAND,NORMAL,SNORM,EXPON, or SEXPON;

ENTRY POINT	FUNCTION
OVFLOW	Calls SPIE, handles fixed point overflows. (Must be called once at start of program.)
INT	Generates integer random numbers.
SINT	Generates shuffled integer random numbers.
RANDOM	Generates single precision floating point (0.0,1.0) random numbers.
SRAND	Generates single precision floating point (0.0,1.0) shuffled random numbers.
NORMAL	Generates single precision floating point normal deviates (μ=0,σ=1).
SNORM	Generates shuffled single precision floating point normal deviates (μ=0,σ=1).
EXPON	Generates single precision floating point exponential deviates (λ=1).
SEXPON	Generates shuffled single precision floating point exponential deviates (λ=1).

TABLE 1.

ENTRY POINT NAMES OF CONTROL SECTION OVFLOW

IX is the starting value of the sequence and may contain <u>any</u> integer number between 1 and 2147483647. This variable should <u>not</u> be altered by the user during the execution of the program, unless it is desired to repeat a sequence of random numbers.

A is either a scalar or vector variable and is the location with a specified dimension into which the random number or numbers are stored (see next parameter). Note that for entry points INT and SINT, this argument should be of INTEGER type.

N is an integer variable or constant designating how many random numbers are to be generated during this call. If N is greater than 1, A above must be a vector dimensioned at least as large as N. If N is equal to 1, then A may be scalar.

Some sample programs are given below:

(1) To generate 1000 consecutive integer random numbers:

```
INTEGER*4   M(1000)

CALL OVFLOW

IX = 1234567

--

--

--

CALL INT (IX, M, 1000)

--

END
```

(2) To generate 25 shuffled single precision floating point normal deviates and scale to mean 10 and standard deviation 5:

```
REAL*4   A(25)

CALL OVFLOW

JJ = 1936748

N = 25

--

--

--
```

```
      CALL SNORM (JJ, A, N)
      DO 1 I = 1,25
      A(I) = A(I)*5.0 + 10.0
    1 CONTINUE
      --
      --
      --
      END
```

(3) To generate one single precision floating point exponential deviate with mean 6:

```
      CALL OVFLOW
      I9 = 98367221
      --
      --
      --
      CALL EXPON (I9, E, 1)
      E = E*6.0
      --
      --
      --
      END
```

A. Implementation. LLRANDOM was designed and coded to run under Operating System/360 (OS). The Assembler Language control section contains a SPIE (Set Program Interrupt Exit) macro instruction which is a part of the OS Supervisor Services. This macro enables LLRANDOM to correct for the fixed point overflows resulting from the division simulation algorithm.

The remainder of the assembly coding is in Basic Assembler Language (BAL), i.e. no other macro calls or supervisor calls. To run LLRANDOM under another operating system for the System/360, an appropriate substitution for the SPIE macro would be necessary.

As currently programmed, LLRANDOM has the following memory requirements:

MODULE	SIZE IN BYTES (DECIMAL)
Assembler CSECT	3571
FORTRAN function RNORTH	1512
FORTRAN function REXPTH	1106
Total memory requirement	6189

The System/360 internal timer is rather crude for timing the execution of programs. The following times are therefore approximate timings for the generation of pseudo-random numbers on a System/360 Model 67-2.

ENTRY POINT	TIME IN MICROSECONDS	
INT	10.7	
SINT	15.7	
RANDOM	15.6	
SRAND	20.0	
NORMAL	57.5	(Polar method takes 349 microseconds)
SNORM	65.8	
EXPON	59.1	(Logarithm method takes 132 microseconds)
SEXPON	68.4	

B. Future Enhancements. The normal and exponential deviate routines in LLRANDOM are patterned after a package, SUPER-DUPER, available from Professor G. Marsaglia at McGill University in Montreal. It is available at the Naval Postgraduate School. Marsaglia uses a different multiplier, A, and modulus, m, in his congruential generator from that used in LLRANDOM and he then exclusive OR's this result with the output of a feedback shift register generator. SUPER-DUPER provides only one deviate per call and does not provide for shuffling the sequence.

The two FORTRAN function subprograms, RNORTH and REXPTH, are taken (with slight modification) directly from SUPER-DUPER. Among the changes to be made to LLRANDOM will be to rewrite RNORTH and REXPTH in System/360 Assembly Language and incorporate them directly into the package.

We have experienced occasions where large-scale simulations have been coded in FORTRAN using double precision variables. The fact that LLRANDOM returns single precision numbers causes some inconvenience. To alleviate this problem we will provide the capability in LLRANDOM to return single precision numbers into double precision variables or arrays. Note that the values returned will still be single precision; however, they will be stored properly into double precision locations.

Finally, additional entry points will be added to provide single precision floating point gamma deviates. Shuffling of the gamma deviates will also be available.

Other enhancements are under consideration.

REFERENCES

1. Lewis, P. A. W., A. S. Goodman and J. M. Miller, Pseudo-Random Number Generator for the System/360, IBM Systems Journal, No. 2, 1969.

2. Payne, W. H., J. R. Rabung and T. P. Bogyo, Coding the Lehmer Pseudo-random Number Generator, Communications of the ACM, Vol. 12, No. 2, February 1969.

3. Knuth, D. E., The Art of Computer Programming, Volume 2: Seminumerical Algorithms, Addison-Wesley, Reading, Mass., 1969.

4. Gaver, D. P. and G. L. Thompson, Programming and Probability Models in Operations Research, Brooks/Cole, Monterey, Calif., 1973.

5. Ahrens, J. H. and U. Dieter, Computer methods for sampling from the exponential and normal distributions, Comm. of the ACM, 15, 873-882, 1972.

6. Lewis, P. A. W., Large-Scale Computer-Aided Statistical Mathematics, Proceedings of the Computer Science and Statistics 6th Annual Symposium on the Interface, Michael E. Tarter (ed.), University of California, Berkeley, 1973.

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
C       REXP TOOTH FUNCTION
        FUNCTION REXPTH(K,IX)
        DIMENSION C(65)
        DATA C/Z40F00000,Z40E10000,Z40D40000,Z40C70000,Z40BB0000,
       $ Z40AF0000,Z40A50000,Z409B0000,Z40910000,Z40890000,Z40800000,
       $ Z4078000C,Z40710000,Z406A0000,Z40640000,Z405E000C,Z40580000,
       $ Z40530000,Z404E0000,Z40490000,Z40440000,Z4040000C,Z403C0000,
       $ Z40390000,Z40350000,Z40320000,Z402F0000,Z402C0000,Z40290000,
       $ Z40270000,Z40240000,Z40220000,Z40200000,Z401E0000,Z401C0000,
       $ Z401A0000,Z40190000,Z40170000,Z40160000,Z40150000,Z40130000,
       $ Z40120000,Z40110000,Z40100000,Z3FF00000,Z3FE00000,Z3FD00000,
       $ Z3FC00000,Z3FB00000,Z3FB00000,Z3FA00000,Z3F90000C,Z3F900000,
       $ Z3F800000,Z3F800000,Z3F700000,Z3F700000,Z3F600000,Z3F600000,
       $ Z3F6C0000,Z3F500000,Z3F500000,Z3F400000,Z3F400000,Z3F400000/
        DATA I1/ZFB4FAA91/
        IF(K.GT.I1)GO TO 5
1       CALL RANDOM(IX,U1,1)
        IF(U1.GT..7917049) GO TO 3
        T=1.-1.235962*U1
        REXPTH=-ALOG(T)
        J=16.*REXPTH+1.
        CALL RANDOM(IX,Z9,1)
        IF(Z9*(.0604*T+.0039).GT.T-C(J)) GO TO 1
        RETURN
3       REXPTH=19.20352*U1-15.20352
        J=16.*REXPTH+1.
        EX=EXP(-REXPTH)
        CALL RANDOM(IX,Z9,1)
        IF(Z9*(.0604*EX+.0039).GT.EX-C(J)) GO TO 1
        RETURN
5       CALL RANDOM(IX,U1,1)
        IF(U1.EQ.0)GO TO 5
        REXPTH=4.-ALOG(U1)
        RETURN
        END
```

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
C RNOR TOOTH FUNCTION
      FUNCTION RNORTH(K,IX)
      DIMENSION C(45)
      DATA C/Z40FD2B5F,Z40FD2B5F,Z40FAA9AD,Z40F5A648,Z40F32496,
     $ Z40EE2131,Z40E69C1A,Z40E198B5,Z40DA139E,Z40D28E87,Z40C887BE,
     $ Z40C102A6,Z40B6FBDD,Z40ACF513,Z40A2EE4A,Z4098E78C,Z40916269,
     $ Z40875BAC,Z407D54D6,Z40734E0D,Z406BC8F6,Z4061C22C,Z405A3D15,
     $ Z4052B7FE,Z404B32E7,Z4043ADD0,Z403C28B9,Z40372554,Z402FA03D,
     $ Z402A9CD8,Z40259973,Z4020960E,Z401E145C,Z401910F7,Z40168F45,
     $ Z40140D93,Z40118BE0,Z3FF0A2E4,Z3FC887BE,Z3FA06C98,Z3F785172,
     $ Z3F785172,Z3F50364C,Z3F50364C,Z3F50364C/
      DATA I1/ZFBC35400/,I2/ZFE79702E/
      IF(K.GT.I1)GO TO 3
      CALL RANDOM(IX,S,1)
      CALL RANDCM(IX,T,1)
      B=AINT(7.*(S+T)+37.*ABS(S-T))
      CALL RANDOM(IX,Z9,1)
      CALL RANDOM(IX,Z8,1)
        X=Z9-Z8
      RNORTH=.0625*(X+SIGN(B,X))
      RETURN
3     IF(K.GT.I2)GO TO 5
4     CALL RANDOM(IX,Z9,1)
      CALL RANDCM(IX,Z8,1)
      IF(Z8.GT.0.50) Z9=-Z9
      RNORTH=2.75*Z9
      J=16.*ABS(RNORTH)+1.
      IF(J-14) 6,6,7
  6   P=(J+J-1)*.1497466E-2
      GO TO 8
  7   P=(89-J-J)*.698817E-3
8     CALL RANDCM(IX,Z9,1)
      IF(Z9.GT.79.78846*(EXP(-.5*RNORTH*RNORTH)
     $     -C(J)-P*(J-16.*ABS(RNORTH)))) GOTO4
      RETURN
5     CALL RANDOM(IX,V,1)
      CALL RANDOM(IX,Z9,1)
      IF(Z9.GT.0.5) V=-V
      IF(V.EQ.0) GO TO 5
      X=SQRT(7.5625-2.*ALOG(ABS(V)))
      CALL RANDOM (IX,Z9,1)
      IF(Z9*X.GT.2.75) GO TO 5
      RNORTH=SIGN(X,V)
      RETURN
      END
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

OVFLOW   CSECT                                                         LLRA0010
         EXTRN RNORTH,REXPTH                                           LLRA0020
         ENTRY INT,SINT,RANDOM,SRAND,NORMAL,SNORM,EXPON,SEXPON         LLRA0030
         USING OVFLOW,R12                                              LLRA0040
         B     12(,R15)           BRANCH AROUND ID                     LLRA0050
         DC    AL1(6)                                                  LLRA0060
         DC    CL6'OVFLOW'                                             LLRA0070
         STM   R14,R12,12(R13) SAVE REGISTERS IN HIGH SAVE AREA        LLRA0080
         LR    R12,R15            ESTABLISH BASE ADDRESS               LLRA0090
         ST    R13,SA+4           SAVE CALLER'S R13                    LLRA0100
         LR    R2,R13                                                  LLRA0110
         LA    R13,SA             NEW SAVE AREA                        LLRA0120
         ST    R13,8(,R2)         STORE WITH CALLING ROUTINE           LLRA0130
*                                                                      LLRA0140
*    ISSUE SPIE TO GET FIXED POINT OVERFLOWS AS WELL AS FCRTRAN        LLRA0150
*    INTERRLPTS.                                                       LLRA0160
*                                                                      LLRA0170
         SPIE  FIXIT,(8,9,12,13,15)                                    LLRA0180
         ST    R1,PICA            SAVE FORTRAN'S PICA ADDRESS          LLRA0190
         L     R13,SA+4           RESTORE CALLER'S R13                 LLRA0200
         LM    R14,R12,12(R13) RESTORE THE REGISTERS                   LLRA0210
         BCR   15,R14             RETURN                               LLRA0220
```

***** NAVAL POSTCRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
*                                                                       LLRA0240
*    SPIE BRINGS US HERE ON INTERRUPTS                                  LLRA0250
*                                                                       LLRA0260
         USING *,R15                                                    LLRA0270
FIXIT    TM    7(R1),X'F7'      WAS IT A FIXED PCINT OVERFLCW?          LLRA0280
         BC    5,FORT           NO, LET FORTRAN'S SPIE HANDLE IT        LLRA0290
         CLC   17(3,R1),AINT+1    TEST WHETHER BASE OF INTERRUPTED      LLRA0300
         BL    0(,R14)               ROUTINE WAS BETWEEN ENTRIES INT AND LLRA0310
         CLC   17(3,R1),ASEXPO+1        SEXPON INCLUSIVE; IF NOT, IGNORE LLRA0320
         BH    0(,R14)                     THE INTERRUPT                LLRA0330
         A     R4,PM2           ADD 2**31-3                             LLRA0340
         AR    R4,R2            ADD 4 MORE TO MAKE 2**31+1 CORRECTION   LLRA0350
         BR    R14              ALL FIXED, CONTINUE                     LLRA0360
FORT     L     R15,PICA         NOT FIXED POINT OVERFLOW, LET FORTRAN   LLRA0370
         L     R15,0(,R15)         EXTENDED ERROR HANDLING RCUTINE      LLRA0380
         FR    R15                    TAKE CARE OF IT                   LLRA0390
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

*                                                                       LLRA0410
*    ENTRY POINT :  INT                                                 LLRA0420
*                                                                       LLRA0430
         CNOP  0,8                                                      LLRA0440
         USING INT,R15          BASE REGISTER                           LLRA0450
INT      B     8(,R15)          BRANCH AROUND ID                        LLRA0460
         DC    AL1(3)                                                   LLRA0470
         DC    CL3'INT'                                                 LLRA0480
         STM   R14,R12,12(R13)  SAVE REGISTERS IN HIGH SAVE AREA        LLRA0490
         ST    R13,SA+4         ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR. LLRA0500
         LR    R2,R13           COPY TO R2                              LLRA0510
         LA    R13,SA           ADDRESS OF LOW SAVE AREA                LLRA0520
         ST    R13,8(,R2)       ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR. LLRA0530
         L     R9,A75           LOAD MULTIPLIER                         LLRA0540
         LA    R2,4             CONSTANT FOR BXLE                       LLRA0550
         LM    R5,R7,0(R1)      ADDRESSES OF THREE ARGUMENTS            LLRA0560
         L     R5,0(,R5)        LOAD STARTING VALUE INTO R5             LLRA0570
         L     R3,0(,R7)        NUMBER OF CONSECUTIVE WORDS TO FILL     LLRA0580
         SLA   R3,2             CONVERT TO BYTES                        LLRA0590
         SR    R6,R2            BACKUP ONE WORD IN CALLER'S ARRAY       LLRA0600
         LR    R7,R2            INITIAL VALUE FOR INDEX REGISTER        LLRA0610
         CNOP  0,8              ALIGN BXLE LOOP FOR SPEED               LLRA0620
L1       MR    R4,R9            FORM PRODUCT OF A AND X(N-1)            LLRA0630
         SLDA  R4,1             R4= REMAINDER ; R5 = QUOTIENT           LLRA0640
         SRL   R5,1               ADD QUOTIENT TO REMAINDER THEREBY     LLRA0650
         AR    R4,R5                SIMULATING DIVISION BY 2**31-1      LLRA0660
         LR    R5,R4            PUT X(N) INTO R5 FOR NEXT GO AROUND     LLRA0670
         ST    R5,0(R7,R6)      STORE IN CALLER'S ARRAY                 LLRA0680
N1       BXLE  R7,R2,L1         LOOP AROUND AGAIN                       LLRA0690
         L     R4,0(,R1)        GET STARTING VALUE ADDRESS AGAIN        LLRA0700
         ST    R5,0(,R4)        STORE AS STARTING VALUE FOR NEXT CALL   LLRA0710
         L     R13,SA+4         RESTORE CALLER'S SAVE AREA POINTER      LLRA0720
         LM    R14,R12,12(R13)  RESTORE THE REGISTERS                   LLRA0730
         BCR   15,R14           RETURN                                  LLRA0740
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

*                                                                                LLRA0760
*    ENTRY POINT : SINT                                                          LLRA0770
*                                                                                LLRA0780
         CNOP  0,8                                                               LLRA0790
         USING SINT,R15         BASE REGISTER                                    LLRA0800
SINT     B     10(0,R15)        BRANCH AROUND ID                                 LLRA0810
         DC    AL1(4)                                                            LLRA0820
         DC    CL4'SINT'                                                         LLRA0830
         STM   R14,R12,12(R13)  SAVE REGISTERS IN HIGH SAVE AREA                 LLRA0840
         ST    R13,SA+4         ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR.        LLRA0850
         LR    R2,R13           COPY TO R2                                       LLRA0860
         LA    R13,SA           ADDRESS OF LOW SAVE AREA                         LLRA0870
         ST    R13,8(,R2)       ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR.        LLRA0880
         L     R9,A75           LOAD MULTIPLIER                                  LLRA0890
         LA    R2,4             CONSTANT FOR BXLE                                LLRA0900
         LM    R5,R7,0(R1)      ADDRESSES OF THREE ARGUMENTS                     LLRA0910
         L     R5,0(,R5)        LOAD STARTING VALUE INTO R5                      LLRA0920
         L     R3,0(,R7)        NUMBER OF CONSECUTIVE WORDS TO FILL              LLRA0930
         SLA   R3,2             CONVERT TO BYTES                                 LLRA0940
         SR    R6,R2            BACKUP ONE WORD IN CALLER'S ARRAY                LLRA0950
         LR    R7,R2            INITIAL VALUE FOR INDEX REGISTER                 LLRA0960
         LA    R8,TABLE         ADDRESS OF SHUFFLING TABLE                       LLRA0970
         L     R1,MASK          INDEX MASK FOR SHUFFLING                         LLRA0980
         CNOP  0,8              ALIGN BXLE LOOP FOR SPEED                        LLRA0990
L2       MR    R4,R9            FORM PRODUCT OF A AND X(N-1)                     LLRA1000
         SLDA  R4,1             R4 = REMAINDER ; R5 = QUOTIENT                   LLRA1010
         SRL   R5,1               ADD QUOTIENT TO REMAINDER THEREBY              LLRA1020
         AR    R4,R5                SIMULATING DIVISION BY 2**31-1               LLRA1030
         LR    R5,R4            PUT X(N) INTO R5 FOR NEXT GO AROUND              LLRA1040
         NR    R4,R1            EXTRACT RIGHT-MOST 7 BITS                        LLRA1050
         SLA   R4,2             CONVERT TO BYTE OFFSET IN TABLE                  LLRA1060
         L     R0,0(R4,R8)      SELECT RANDOM TABLE VALUE                        LLRA1070
         ST    R5,0(R4,R8)      REPLACE TABLE VALUE WITH X(N)                    LLRA1080
         ST    R0,0(R7,R6)      RANDOM TABLE VALUE TO CALLER'S ARRAY             LLRA1090
N2       BXLE  R7,R2,L2         LOOP AROUND AGAIN                                LLRA1100
         L     R13,SA+4         RESTORE CALLER'S SAVE AREA POINTER               LLRA1110
         L     R1,24(,R13)      GET ARGUMENT LIST POINTER AGAIN                  LLRA1120
         L     R4,0(,R1)        GET STARTING VALUE ADDRESS AGAIN                 LLRA1130
         ST    R5,0(,R4)        STORE AS STARTING VALUE FOR NEXT CALL            LLRA1140
         LM    R14,R12,12(R13)  RESTORE THE REGISTERS                            LLRA1150
         BCR   15,R14           RETURN                                           LLRA1160
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

*                                                                    LLRA1180
*     ENTRY PCINT :  RANDOM                                          LLRA1190
*                                                                    LLRA1200
         CNOP   0,8                                                  LLRA1210
         USING  RANDOM,R15         BASE REGISTER                     LLRA1220
RANDOM   B      12(,R15)           BRANCH AROUND ID                  LLRA1230
         DC     AL1(6)                                               LLRA1240
         DC     CL6'RANDOM'                                          LLRA1250
         STM    R14,R12,12(R13)  SAVE REGISTERS IN HIGH SAVE AREA    LLRA1260
         ST     R13,SA+4           ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR. LLRA1270
         LR     R2,R13             COPY TO R2                        LLRA1280
         LA     R13,SA             ADDRESS OF LOW SAVE AREA          LLRA1290
         ST     R13,8(,R2)         ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR. LLRA1300
         LM     R9,R11,A75         LOAD MULTIPLIER AND NORMALIZATION CONST. LLRA1310
         LA     R2,4               CONSTANT FOR BXLE                 LLRA1320
         LM     R5,R7,0(R1)        ADDRESSES OF THREE ARGUMENTS      LLRA1330
         L      R5,0(,R5)          LOAD STARTING VALUE INTO R5       LLRA1340
         L      R3,0(,R7)          NUMBER OF CONSECUTIVE WORDS TO FILL LLRA1350
         SLA    R3,2               CONVERT TO BYTES                  LLRA1360
         SR     R6,R2              BACKUP ONE WORD IN CALLER'S ARRAY LLRA1370
         LR     R7,R2              INITIAL VALUE FOR INDEX REGISTER  LLRA1380
         SDR    FR0,FR0            CLEAR FLOATING POINT REGISTER 0   LLRA1390
         LA     R12,N3             ADDRESS OF BXLE INSTRUCTION       LLRA1400
         LA     R13,M3             ADDRESS OF NORMALIZATION ROUTINE  LLRA1410
         CNOP   0,8                ALIGN BXLE LOOP FOR SPEED         LLRA1420
L3       MR     R4,R9              FORM PRODUCT OF A AND X(N-1)      LLRA1430
         SLDA   R4,1               R4 = REMAINDER ; R5 = QUOTIENT    LLRA1440
         SRL    R5,1                 ADD QUOTIENT TO REMAINDER THEREBY LLRA1450
         AR     R4,R5                  SIMULATING DIVISION BY 2**31-1 LLRA1460
         LR     R5,R4              PUT X(N) INTO R5 FOR NEXT GO AROUND LLRA1470
         SRL    R4,7               MAKE ROOM FOR THE EXPONENT        LLRA1480
         OR     R4,R10             OR ON THE EXPONENT                LLRA1490
         ST     R4,0(R7,R6)        STORE IN CALLER'S ARRAY           LLRA1500
         CR     R4,R11             DID IT NEED NORMALIZATION?        LLRA1510
         BCR    4,R13              YES, GO NORMALIZE IT              LLRA1520
N3       BXLE   R7,R2,L3           LOOP AROUND AGAIN                 LLRA1530
         L      R4,0(,R1)          GET STARTING VALUE ADDRESS AGAIN  LLRA1540
         ST     R5,0(,R4)          STORE AS STARTING VALUE FOR NEXT CALL LLRA1550
         L      R13,SA+4           RESTORE CALLER'S SAVE AREA POINTER LLRA1560
         LM     R14,R12,12(R13)  RESTORE THE REGISTERS               LLRA1570
         BCR    15,R14             RETURN                            LLRA1580
M3       LE     FR2,0(R7,R6)       LOAD INTO FLOATING POINT REGISTER 2 LLRA1590
         AER    FR2,FR0            ADD ZERO AND NORMALIZE            LLRA1600
         STE    FR2,0(R7,R6)       STORE BACK NORMALIZED             LLRA1610
         BR     R12                CONTINUE THE BXLE LOOP            LLRA1620
```

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
*
*     ENTRY POINT :  SRAND
*
         CNOP  0,8
         USING SRAND,R15         BASE REGISTER
SRAND    B     10(,R15)          BRANCH AROUND ID
         DC    AL1(5)
         DC    CL5'SRAND'
         STM   R14,R12,12(R13)  SAVE REGISTERS IN HIGH SAVE AREA
         ST    R13,SA+4          ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR.
         LR    R2,R13            COPY TO R2
         LA    R13,SA            ADDRESS OF LOW SAVE AREA
         ST    R13,8(,R2)        ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR.
         LM    R9,R11,A75        LOAD MULTIPLIER AND NORMALIZATION CONST.
         LA    R2,4              CONSTANT FOR BXLE
         LM    R5,R7,0(R1)       ADDRESSES OF THREE ARGUMENTS
         L     R5,0(,R5)         LOAD STARTING VALUE INTO R5
         L     R3,0(,R7)         NUMBER OF CONSECUTIVE WORCS TO FILL
         SLA   R3,2              CONVERT TO BYTES
         SR    R6,R2             BACKUP ONE WORD IN CALLER'S ARRAY
         LR    R7,R2             INITIAL VALUE FOR INDEX REGISTER
         SUR   FR0,FR0           CLEAR FLOATING POINT REGISTER 0
         LA    R12,N4            ADDRESS OF BXLE INSTRUCTICN
         LA    R13,M4            ADDRESS OF NORMALIZATION RCUTINE
         LA    R8,TABLE          ADDRESS OF SHUFFLING TABLE
         L     R1,MASK           INDEX MASK FOR SHUFFLING
         CNOP  0,8               ALIGN BXLE LOOP FOR SPEED
L4       MR    R4,R9             FORM PRODUCT OF A AND X(N-1)
         SLDA  R4,1              R4 = REMAINDER ; R5 = QUOTIENT
         SRL   R5,1                ADD QUOTIENT TO REMAINDER THEREBY
         AR    R4,R5                 SIMULATING DIVISION BY 2**31-1
         LR    R5,R4             PUT X(N) INTO R5 FOR NEXT GC AROUND
         NR    R4,R1             EXTRACT RIGHT-MOST 7 BITS
         SLA   R4,2              CONVERT TO BYTE OFFSET IN TABLE
         L     R0,0(R4,R8)       SELECT RANDOM TABLE VALUE
         ST    R5,0(R4,R8)       REPLACE TABLE VALUE WITH X(N)
         SRL   R0,7              MAKE ROOM FOR THE EXPONENT
         OR    R0,R10            OR ON THE EXPONENT
         ST    R0,0(R7,R6)       STORE IN CALLER'S ARRAY
         CR    R0,R11            DID IT NEED NORMALIZATION ?
         BCR   4,R13             YES, GO NORMALIZE IT
N4       BXLE  R7,R2,L4          LOCP AROUND AGAIN
         L     R13,SA+4          RESTORE CALLER'S SAVE AREA POINTER
         L     R1,24(,R13)       GET ARGUMENT LIST POINTER AGAIN
         L     R4,0(,R1)         GET STARTING VALUE ADDRESS AGAIN
         ST    R5,0(,R4)         STORE AS STARTING VALUE FOR NEXT CALL
         LM    R14,R12,12(R13)  RESTORE THE REGISTERS
         BCR   15,R14            RETURN
M4       LE    FR2,0(R7,R6)      LOAD INTO FLOATING POINT REGISTER 2
```

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
         AER    FR2,FR0          ADD ZERO AND NORMALIZE                    LLRA2130
         STE    FR2,0(R7,R6)     STORE BACK NORMALIZED                     LLRA2140
         BR     R12              CONTINUE THE BXLE LOOP                    LLRA2150
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

*
*    ENTRY PCINT :  NORMAL
*
         CNOP  0,8
         USING NORMAL,R15         BASE REGISTER
NORMAL   B     12(,R15)           BRANCH AROUND ID
         DC    AL1(6)
         DC    CL6'NORMAL'
         STM   R14,R12,12(R13)  SAVE REGISTERS IN HIGH SAVE AREA
         ST    R13,SA2+4          ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR.
         LR    R2,R13             COPY TO R2
         LA    R13,SA2            ADDRESS OF LOW SAVE AREA
         ST    R13,8(,R2)         ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR.
         LM    R9,R11,A75N        LOAD MULTIPLIER,EXPONENT,AND TEST MASK
         LA    R2,4               CONSTANT FOR BXLE
         LM    R5,R7,0(R1)        ADDRESSES OF THREE ARGUMENTS
         L     R5,0(,R5)          LOAD STARTING VALUE INTO R5
         L     R3,0(,R7)          NUMBER OF CONSECUTIVE WORDS TO FILL
         SLA   R3,2               CONVERT TO BYTES
         SR    R6,R2              BACKUP ONE WORD IN CALLER'S ARRAY
         LR    R7,R2              INITIAL VALUE FOR INDEX REGISTER
         LA    R13,ATBLE          ADDRESS OF TABLE OF CONSTANTS
         LA    R12,N5             ADDRESS OF BXLE
         CNOP  0,8                ALIGN BXLE LOOP FOR SPEED
L5       MR    R4,R9              FORM PRODUCT OF A AND X(N-1)
         SLDA  R4,1               R4 = REMAINDER ; R5 = QUOTIENT
         SRL   R5,1                 ADD QUOTIENT TO REMAINDER THEREBY
         AR    R4,R5                  SIMULATING DIVISION BY 2**31-1
         LR    R5,R4              PUT X(N) INTO R5
         LR    R0,R5              COPY R5 INTO R0 FOR NOW
         NR    R4,R11             SHOULD WE MAKE IT NEGATIVE ?
         BC    8,F1               POSITIVE, KEEP GOING
         LNR   R5,R5              MAKE R5 TRUE NEGATIVE
         SLR   R4,R4              CLEAR R4 TO ZERO
F1       CL    R5,C1              R5 LESS THAN X'68000000' ?
         BC    11,F2              NO
         SLDL  R4,8               SHIFT FIRST 8 BITS OF R5 INTO R4 AS INDEX
         IC    R4,0(R4,R13)       OBTAIN CONSTANT FROM TABLE
         STC   R4,PWRD+1          STORE IN SECOND BYTE OF PWRD
         SRL   R5,8               SHIFT REMAINING 24 BITS RIGHT THEN OR ON
         ALR   R5,R10             EXPONENT TO MAKE  .(24 BITS)/16
         ST    R5,0(R7,R6)        STORE IN CALLER'S ARRAY
         LE    FR0,PWRD           LOAD CHARACTERISTIC TO FLOATING POINT
         AE    FR0,0(R7,R6)       REGISTER 0 AND ADD FRACTION
         STE   FR0,0(R7,R6)       STORE NORMAL DEVIATE IN CALLER'S ARRAY
         LR    R5,R0              COPY BACK TO R5 FOR NEXT GO AROUND
         BR    R12                GO TO BXLE AND CONTINUE
F2       CL    R5,C2              R5 LESS THAN X'D0000000' ?
         BC    11,F3              NO
```

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
         SLDL  R4,8             SHIFT FIRST 8 BITS OF R5 INTO R4 AS INDEX LLRA2660
         SL    R4,C1M           SUBTRACT 68                               LLRA2670
         IC    R4,0(R4,R13)     OBTAIN CONSTANT FROM TABLE                LLRA2680
         STC   R4,NWRD+1        STORE IN SECOND BYTE OF NWRD              LLRA2690
         SRL   R5,8             SHIFT REMAINING 24 BITS RIGHT THEN OR ON  LLRA2700
         ALR   R5,R10           EXPONENT TO MAKE .(24 BITS)/16            LLRA2710
         ST    R5,0(R7,R6)      STORE IN CALLER'S ARRAY                   LLRA2720
         LE    FR0,NWRD         LOAD CHARACTERISTIC TO FLOATING POINT     LLRA2730
         SE    FR0,0(R7,R6)     REGISTER 0 AND SUBTRACT FRACTION          LLRA2740
         STE   FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY    LLRA2750
         LR    R5,R0            COPY BACK TO R5 FOR NEXT GO AROUND        LLRA2760
         BR    R12              GO TO BXLE AND CONTINUE                   LLRA2770
F3       CL    R5,C3            R5 LESS THAN X'E2F00000' ?                LLRA2780
         BC    11,F4            NO                                        LLRA2790
         SLDL  R4,12            SHIFT FIRST 12 BITS OF R5 INTO R4         LLRA2800
         SL    R4,C2M           SUBTRACT CE8                              LLRA2810
         IC    R4,0(R4,R13)     OBTAIN CONSTANT FROM TABLE                LLRA2820
         STC   R4,PWRD+1        STORE IN SECOND BYTE OF PWRD              LLRA2830
         SRL   R5,8             SHIFT REMAINING 20 BITS RIGHT THEN OR ON  LLRA2840
         ALR   R5,R10           EXPONENT TO MAKE .(20 BITS)/16            LLRA2850
         ST    R5,0(R7,R6)      STORE IN CALLER'S ARRAY                   LLRA2860
         LE    FR0,PWRD         LOAD CHARACTERISTIC TO FLOATING POINT     LLRA2870
         AE    FR0,0(R7,R6)     REGISTER 0 AND ADD FRACTION               LLRA2880
         STE   FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY    LLRA2890
         LR    R5,R0            COPY BACK TO R5 FOR NEXT GO AROUND        LLRA2900
         BR    R12              GO TO BXLE AND CONTINUE                   LLRA2910
F4       CL    R5,C4            R5 LESS THAN X'F5E00000' ?                LLRA2920
         BC    11,F5            NO                                        LLRA2930
         SLDL  R4,12            SHIFT FIRST 12 BITS OF R5 INTO R4         LLRA2940
         SL    R4,C3M           SUBTRACT E17                              LLRA2950
         IC    R4,0(R4,R13)     OBTAIN CONSTANT FROM TABLE                LLRA2960
         STC   R4,NWRD+1        STORE AS SECOND BYTE OF NWRD              LLRA2970
         SRL   R5,8             SHIFT REMAINING 20 BITS RIGHT THEN OR ON  LLRA2980
         ALR   R5,R10           EXPONENT TO MAKE .(20 BITS)/16            LLRA2990
         ST    R5,0(R7,R6)      STORE IN CALLER'S ARRAY                   LLRA3000
         LE    FR0,NWRD         LOAD CHARACTERISTIC TO FLOATING POINT     LLRA3010
         SE    FR0,0(R7,R6)     REGISTER 0 AND SUBTRACT FRACTION          LLRA3020
         STE   FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY    LLRA3030
         LR    R5,R0            COPY BACK TO R5 FOR NEXT GO AROUND        LLRA3040
         BR    R12              GO TO BXLE AND CONTINUE                   LLRA3050
F5       ST    R5,FWRD          STORE R5 IN ARGUMENT LIST                 LLRA3060
         ST    R0,XWRD          PASS STARTING VALUE                       LLRA3070
         LA    R13,SA2          LOAD LOW SAVE AREA POINTER                LLRA3080
         LA    R1,FLIST         ARGUMENT LIST FOR CALL TO RNORTH          LLRA3090
         LR    R8,R15           COPY BASE REGISTER FOR BALR LINKAGE       LLRA3100
         L     R15,ARNOR        ADDRESS OF FUNCTION SUBROUTINE RNORTH     LLRA3110
         BALR  R14,R15          BRANCH TO RNORTH                          LLRA3120
         LR    R15,R8           RESTORE BASE REGISTER                     LLRA3130
         STE   FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY    LLRA3140
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

         L     R5,XWRD           NEW STARTING VALUE                      LLRA3150
         LA    R13,ATBLE         RESTORE R13 TO TABLE OF CCNSTANTS       LLRA3160
N5       BXLE  R7,R2,L5          LOOP AROUND AGAIN                       LLRA3170
         L     R13,SA2+4         RESTORE HIGH SAVE AREA POINTER          LLRA3180
         L     R1,24(,13)        GET ARGUMENT LIST POINTER AGAIN         LLRA3190
         L     R4,0(,R1)         GET STARTING VALUE ADDRESS AGAIN        LLRA3200
         ST    R5,0(,R4)         STORE AS STARTING VALUE FCR NEXT CALL   LLRA3210
         LM    R14,R12,12(R13) RESTORE THE REGISTERS                     LLRA3220
         BCR   15,R14            RETURN                                  LLRA3230
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

*                                                                         LLRA3250
*    ENTRY POINT : SNORM                                                  LLRA3260
*                                                                         LLRA3270
         CNOP  0,8                                                        LLRA3280
         USING SNORM,R15       BASE REGISTER                              LLRA3290
SNORM    B     10(,R15)        BRANCH AROUND ID                           LLRA3300
         DC    AL1(5)                                                     LLRA3310
         DC    CL5'SNORM'                                                 LLRA3320
         STM   R14,R12,12(R13) SAVE REGISTERS IN HIGH SAVE AREA           LLRA3330
         ST    R13,SA2+4       ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR.  LLRA3340
         LR    R2,R13          COPY TO R2                                 LLRA3350
         LA    R13,SA2         ADDRESS OF LOW SAVE AREA                   LLRA3360
         ST    R13,8(,R2)      ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR.  LLRA3370
         LM    R9,R11,A75N     LOAD MULTIPLIER,EXPONENT,AND TEST MASK     LLRA3380
         LA    R2,4            CONSTANT FOR BXLE                          LLRA3390
         LM    R5,R7,0(R1)     ADDRESSES OF THREE ARGUMENTS               LLRA3400
         L     R5,0(,R5)       LOAD STARTING VALUE INTO R5                LLRA3410
         L     R3,0(,R7)       NUMBER OF CONSECUTIVE WORDS TO FILL        LLRA3420
         SLA   R3,2            CONVERT TO BYTES                           LLRA3430
         SR    R6,R2           BACKUP ONE WORD IN CALLER'S ARRAY          LLRA3440
         LR    R7,R2           INITIAL VALUE FOR INDEX REGISTER           LLRA3450
         LA    R13,ATBLE       ADDRESS OF TABLE OF CONSTANTS              LLRA3460
         LA    R8,TABLE        ADDRESS OF SHUFFLING TABLE                 LLRA3470
         LA    R12,N6          ADDRESS OF BXLE                            LLRA3480
         L     R1,MASK         INDEX MASK FOR SHUFFLING                   LLRA3490
         CNOP  0,8             ALIGN BXLE LOOP FOR SPEED                  LLRA3500
L6       MR    R4,R9           FORM PRODUCT OF A AND X(N-1)               LLRA3510
         SLDA  R4,1            R4 = REMAINDER ; R5= QUOTIENT              LLRA3520
         SRL   R5,1              ADD QUOTIENT TO REMAINDER THEREBY        LLRA3530
         AR    R4,R5               SIMULATING DIVISION BY 2**31-1         LLRA3540
         LR    R5,R4           PUT X(N) INTO R5                           LLRA3550
         NR    R4,R1           EXTRACT RIGHT-MOST 7 BITS                  LLRA3560
         SLA   R4,2            CONVERT TO BYTE OFFSET IN TABLE            LLRA3570
         L     F0,0(R4,R8)     SELECT RANDOM TABLE VALUE                  LLRA3580
         ST    R5,0(R4,R8)     REPLACE TABLE VALUE WITH X(N)              LLRA3590
         XR    R0,R5           EXCHANGE R0 AND R5                         LLRA3600
         XR    R5,R0             BY EXCLUSIVE OR'ING                      LLRA3610
         XR    R0,R5               THEM WITH EACH OTHER                   LLRA3620
         NR    R4,R11          SHOULD WE MAKE IT NEGATIVE ?               LLRA3630
         BC    8,F1S           POSITIVE, KEEP GOING                       LLRA3640
         LNR   R5,R5           MAKE R5 TRUE NEGATIVE                      LLRA3650
         SLR   R4,R4           CLEAR R4 TO ZERO                           LLRA3660
F1S      CL    R5,C1           R5 LESS THAN X'68000000' ?                 LLRA3670
         BC    11,F2S          NO                                         LLRA3680
         SLDL  R4,8            SHIFT FIRST 8 BITS OF R5 INTO R4 AS INDEX  LLRA3690
         IC    R4,0(R4,R13)    OBTAIN CONSTANT FROM TABLE                 LLRA3700
         STC   R4,PWRD+1       STORE IN SECOND BYTE OF PWRD               LLRA3710
         SRL   R5,8            SHIFT REMAINING 24 BITS RIGHT THEN OR ON   LLRA3720
         ALR   R5,R10          EXPONENT TO MAKE  .(24 BITS)/16            LLRA3730
```

```
         ST     R5,0(R7,R6)      STORE IN CALLER'S ARRAY                      LLRA3740
         LE     FR0,PWRD         LOAD CHARACTERISTIC TO FLOATING POINT        LLRA3750
         AE     FR0,0(R7,R6)     REGISTER 0 AND ADD FRACTION                  LLRA3760
         STE    FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY       LLRA3770
         LR     R5,R0            COPY BACK TO R5 FOR NEXT GO AROUND           LLRA3780
         BR     R12              GO TO BXLE AND CONTINUE                      LLRA3790
F2S      CL     R5,C2            R5 LESS THAN X'D0000000' ?                   LLRA3800
         BC     11,F3S           NO                                           LLRA3810
         SLDL   R4,8             SHIFT FIRST 8 BITS OF R5 INTO R4 AS INDEX    LLRA3820
         SL     R4,C1M           SUBTRACT 68                                  LLRA3830
         IC     R4,0(R4,R13)     OBTAIN CONSTANT FROM TABLE                   LLRA3840
         STC    R4,NWRD+1        STORE IN SECOND BYTE OF NWRD                 LLRA3850
         SRL    R5,8             SHIFT REMAINING 24 BITS RIGHT THEN OR ON     LLRA3860
         ALR    R5,R10           EXPONENT TO MAKE .(24 BITS)/16               LLRA3870
         ST     R5,0(R7,R6)      STORE IN CALLER'S ARRAY                      LLRA3880
         LE     FR0,NWRD         LOAD CHARACTERISTIC TO FLOATING POINT        LLRA3890
         SE     FR0,0(R7,R6)     REGISTER 0 AND SUBTRACT FRACTION             LLRA3900
         STE    FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY       LLRA3910
         LR     R5,R0            COPY BACK TO R5 FOR NEXT GO AROUND           LLRA3920
         BR     R12              GO TO BXLE AND CONTINUE                      LLRA3930
F3S      CL     R5,C3            R5 LESS THAN X'E2F00000' ?                   LLRA3940
         BC     11,F4S           NO                                           LLRA3950
         SLDL   R4,12            SHIFT FIRST 12 BITS OF R5 INTO R4            LLRA3960
         SL     R4,C2M           SUBTRACT CE8                                 LLRA3970
         IC     R4,0(R4,R13)     OBTAIN CONSTANT FROM TABLE                   LLRA3980
         STC    R4,PWRD+1        STORE IN SECOND BYTE OF PWRD                 LLRA3990
         SRL    R5,8             SHIFT REMAINING 20 BITS RIGHT THEN OR ON     LLRA4000
         ALR    R5,R10           EXPONENT TO MAKE .(20 BITS)/16               LLRA4010
         ST     R5,0(R7,R6)      STORE IN CALLER'S ARRAY                      LLRA4020
         LE     FR0,PWRD         LOAD CHARACTERISTIC TO FLOATING POINT        LLRA4030
         AE     FR0,0(R7,R6)     REGISTER 0 AND ADD FRACTION                  LLRA4040
         STE    FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY       LLRA4050
         LR     R5,R0            COPY BACK TO R5 FOR NEXT GO AROUND           LLRA4060
         BR     R12              GO TO BXLE AND CONTINUE                      LLRA4070
F4S      CL     R5,C4            R5 LESS THAN X'F5E00000' ?                   LLRA4080
         BC     11,F5S           NO                                           LLRA4090
         SLDL   R4,12            SHIFT FIRST 12 BITS OF R5 INTO R4            LLRA4100
         SL     R4,C3M           SUBTRACT E17                                 LLRA4110
         IC     R4,0(R4,R13)     OBTAIN CONSTANT FROM TABLE                   LLRA4120
         STC    R4,NWRD+1        STORE AS SECOND BYTE OF NWRD                 LLRA4130
         SRL    R5,8             SHIFT REMAINING 20 BITS RIGHT THEN OR ON     LLRA4140
         ALR    R5,R10           EXPONENT TO MAKE .(20 BITS)/16               LLRA4150
         ST     R5,0(R7,R6)      STORE IN CALLER'S ARRAY                      LLRA4160
         LE     FR0,NWRD         LOAD CHARACTERISTIC TO FLOATING POINT        LLRA4170
         SE     FR0,0(R7,R6)     REGISTER 0 AND SUBTRACT FRACTION             LLRA4180
         STE    FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY       LLRA4190
         LR     R5,R0            COPY BACK TO R5 FOR NEXT GO AROUND           LLRA4200
         BR     R12              GO TO BXLE AND CONTINUE                      LLRA4210
F5S      ST     R5,FWRD          STORE R5 IN ARGUMENT LIST                    LLRA4220
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

         ST    R0,XWRD          PASS STARTING VALUE                       LLRA4230
         LA    R13,SA2          LOAD LOW SAVE AREA POINTER                LLRA4240
         LA    R1,FLIST         ARGUMENT LIST FOR CALL TO RNORTH          LLRA4250
         LR    R8,R15           COPY BASE FOR BALR LINKAGE                LLRA4260
         L     R15,ARNOR        ADDRESS OF FUNCTION SUBROUTINE RNORTH     LLRA4270
         BALR  R14,R15          BRANCH TO RNORTH                          LLRA4280
         LR    R15,R8           RESTORE BASE REGISTER                     LLRA4290
         STE   FR0,0(R7,R6)     STORE NORMAL DEVIATE IN CALLER'S ARRAY    LLRA4300
         L     R5,XWRD          NEW STARTING VALUE                        LLRA4310
         LA    R13,ATBLE        RESTORE R13 TO TABLE OF CONSTANTS         LLRA4320
         LA    R8,TABLE         RESTORE R8 TO ADDRESS OF SHUFFLING TABLE  LLRA4330
         L     R1,MASK          RESTORE R1 TO INDEX MASK                  LLRA4340
N6       BXLE  R7,R2,L6         LOOP AROUND AGAIN                         LLRA4350
         L     R13,SA2+4        RESTORE HIGH SAVE AREA POINTER            LLRA4360
         L     R1,24(,R13)      GET ARGUMENT LIST POINTER AGAIN           LLRA4370
         L     R4,0(,R1)        GET STARTING VALUE ADDRESS AGAIN          LLRA4380
         ST    R5,0(,R4)        STORE AS STARTING VALUE FOR NEXT CALL     LLRA4390
         LM    R14,R12,12(R13)  RESTORE THE REGISTERS                     LLRA4400
         BCR   15,R14           RETURN                                    LLRA4410
```

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
*
*     ENTRY POINT :  EXPON
*
          CNOP   0,8
          USING  EXPON,R15         BASE REGISTER
EXPON     B      10(,R15)          BRANCH AROUND ID
          DC     AL1(5)
          DC     CL5'EXPON'
          STM    R14,R12,12(R13)  SAVE REGISTERS IN HIGH SAVE AREA
          ST     R13,SA2+4         ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR.
          LR     R2,R13            COPY TO R2
          LA     R13,SA2           ADDRESS OF LOW SAVE AREA
          ST     R13,8(,R2)        ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR.
          LM     R9,R11,A75N       LOAD MULTIPLIER,EXPONENT,AND TEST MASK
          LA     R2,4              CONSTANT FOR BXLE
          LM     R5,R7,0(R1)       ADDRESSES OF THREE ARGUMENTS
          L      R5,0(,R5)         LOAD STARTING VALUE INTO R5
          L      R3,0(,R7)         NUMBER OF CONSECUTIVE WORDS TO FILL
          SLA    R3,2              CONVERT TO BYTES
          SR     R6,R2             BACKUP ONE WORD IN CALLER'S ARRAY
          LR     R7,R2             INITIAL VALUE FOR INDEX REGISTER
          LA     R13,BTBLE         ADDRESS OF TABLE OF CONSTANTS
          LA     R12,N7            ADDRESS OF BXLE
          CNOP   0,8               ALIGN BXLE LOOP FOR SPEED
L7        MR     R4,R9             FORM PRODUCT OF A AND X(N-1)
          SLDA   R4,1              R4 = REMAINDER ; R5 = QUOTIENT
          SRL    R5,1                ADD QUOTIENT TO REMAINDER THEREBY
          AR     R4,R5                 SIMULATING DIVISION BY 2**31-1
          LR     R5,R4             PUT X(N) INTO R5
          LR     R0,R5             COPY R5 INTO R0 FOR NOW
          NR     R4,R11            SHOULD WE MAKE IT NEGATIVE ?
          BC     8,E1              POSITIVE, KEEP GOING
          LNR    R5,R5             MAKE R5 TRUE NEGATIVE
          SLR    R4,R4             CLEAR R4 TO ZERO
E1        CL     R5,D1             R5 LESS THAN X'D5000000' ?
          BC     11,E2             NO
          SLDL   R4,8              SHIFT FIRST 8 BITS OF R5 INTO R4 AS INDEX
          IC     R4,0(R4,R13)      OBTAIN CONSTANT FROM TABLE
          STC    R4,PWRD+1         STORE IN SECOND BYTE OF PWRD
          SRL    R5,8              SHIFT REMAINING 24 BITS RIGHT THEN OR ON
          ALR    R5,R10            EXPONENT TO MAKE .(24 BITS)/16
          ST     R5,0(R7,R6)       STORE IN CALLER'S ARRAY
          LE     FR0,PWRD          LOAD CHARACTERISTIC TO FLOATING POINT
          AE     FR0,0(R7,R6)      REGISTER 0 AND ADD FRACTION
          STE    FR0,0(R7,R6)      STORE EXPONENTIAL DEVIATE IN ARRAY
          LR     R5,R0             COPY BACK TO R5 FOR NEXT GO AROUND
          BR     R12               GO TO BXLE AND CONTINUE
E2        CL     R5,D2             R5 LESS THAN X'F1700000' ?
          BC     11,E3             NO
```

```
***** NAVAL PCSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

        SLDL  R4,12             SHIFT FIRST 12 BITS OF R5 INTC R4              LLRA4920
        SL    R4,DIM            SUBTRACT CFF                                   LLRA4930
        IC    R4,0(R4,R13)      OBTAIN CONSTANT FRCM TABLE                     LLRA4940
        STC   R4,PWRD+1         STORE AS SECOND BYTE OF PWRD                   LLRA4950
        SRL   R5,8              SHIFT REMAINING 20 BITS RIGHT THEN OR ON       LLRA4960
        ALR   R5,R10            EXPONENT TO MAKE .(20 BITS)/16                 LLRA4970
        ST    R5,0(R7,R6)       STORE IN CALLER'S ARRAY                        LLRA4980
        LE    FR0,PWRD          LOAD CHARACTERISTIC TO FLCATING POINT          LLRA4990
        AE    FR0,0(R7,R6)      REGISTER 0 AND ADD FRACTION                    LLRA5000
        STE   FR0,0(R7,R6)      STORE EXPONENTIAL DEVIATE IN ARRAY             LLRA5010
        LR    R5,R0             COPY BACK TO R5 FOR NEXT GC ARCUND             LLRA5020
        BR    R12               GO TO BXLE AND CONTINUE                        LLRA5030
E3      ST    R5,EWRD           STCRE R5 IN ARGUMENT LIST                      LLRA5040
        ST    R0,XWRD           PASS STARTING VALUE                            LLRA5050
        LA    R13,SA2           LCAD LOW SAVE AREA POINTER                     LLRA5060
        LA    R1,ELIST          ARGUMENT LIST FOR CALL TC REXPTH               LLRA5070
        LR    R8,R15            COPY BASE REGISTER FOR BALR LINKAGE            LLRA5080
        L     R15,AREXP         ADDRESS OF FUNCTION SUBROUTINE REXPTH          LLRA5090
        BALR  R14,R15           BRANCH TO REXPTH                               LLRA5100
        LR    R15,R8            RESTORE BASE REGISTER                          LLRA5110
        STE   FR0,0(R7,R6)      STORE EXPONENTIAL DEVIATE IN ARRAY             LLRA5120
        L     R5,XWRD           NEW STARTING VALUE                             LLRA5130
        LA    R13,BTBLE         RESTORE R13 TO TABLE OF CONTENTS               LLRA5140
N7      BXLE  R7,R2,L7          LOOP AROUND AGAIN                              LLRA5150
        L     R13,SA2+4         RESTORE HIGH SAVE AREA POINTER                 LLRA5160
        L     R1,24(,R13)       GET ARGUMENT LIST POINTER AGAIN                LLRA5170
        L     R4,0(,R1)         GET STARTING VALUE ADDRESS AGAIN               LLRA5180
        ST    R5,0(,R4)         STORE AS STARTING VALUE FCR NEXT CALL          LLRA5190
        LM    R14,R12,12(R13) RESTCRE THE REGISTERS                            LLRA5200
        BCR   15,R14            RETURN                                         LLRA5210
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

*
*     ENTRY POINT :  SEXPON
*
         CNOP  0,8
         USING SEXPON,R15         BASE REGISTER
SEXPON   B     12(,R15)           BRANCH AROUND ID
         DC    AL1(6)
         DC    CL6'SEXPON'
         STM   R14,R12,12(R13)  SAVE REGISTERS IN HIGH SAVE AREA
         ST    R13,SA2+4          ADDRESS OF HIGH SAVE AREA IN LOW SAVE AR.
         LR    R2,R13             COPY TO R2
         LA    R13,SA2            ADDRESS OF LOW SAVE AREA
         ST    R13,8(,R2)         ADDRESS OF LOW SAVE AREA IN HIGH SAVE AR.
         LM    R9,R11,A75N        LOAD MULTIPLIER,EXPONENT,AND TEST MASK
         LA    R2,4               CONSTANT FOR BXLE
         LM    R5,R7,0(R1)        ADDRESSES OF THREE ARGUMENTS
         L     R5,0(,R5)          LOAD STARTING VALUE IN R5
         L     R3,0(,R7)          NUMBER OF CONSECUTIVE WORDS TO FILL
         SLA   R3,2               CONVERT TO BYTES
         SR    R6,R2              BACKUP ONE WORD IN CALLER'S ARRAY
         LR    R7,R2              INITIAL VALUE FOR INDEX REGISTER
         LA    R13,BTBLE          ADDRESS OF TABLE OF CONSTANTS
         LA    R8,TABLE           ADDRESS OF SHUFFLING TABLE
         LA    R12,N8             ADDRESS OF BXLE
         L     R1,MASK            INDEX MASK FOR SHUFFLING
         CNOP  0,8                ALIGN BXLE LOOP FOR SPEED
L8       MR    R4,R9              FORM PRODUCT OF A AND X(N-1)
         SLDA  R4,1               R4 = QUOTIENT ; R5 = REMAINDER
         SRL   R5,1                 ADD QUOTIENT TO REMAINDER THEREBY
         AR    R4,R5                  SIMULATING DIVISION BY 2**31-1
         LR    R5,R4              PUT X(N) INTO R5
         NR    R4,R1              EXTRACT RIGHT-MOST 7 BITS
         SLA   R4,2               CONVERT TO BYTE OFFSET IN TABLE
         L     R0,0(R4,R8)        SELECT RANDOM TABLE VALUE
         ST    R5,0(R4,R8)        REPLACE TABLE VALUE WITH X(N)
         XR    R0,R5              EXCHANGE R0 AND R5
         XR    R5,R0                BY EXCLUSIVE OR'ING
         XR    R0,R5                  THEM WITH EACH OTHER
         NR    R4,R11             SHOULD WE MAKE IT NEGATIVE ?
         BC    8,E1S              POSITIVE, KEEP GOING
         LNR   R5,R5              MAKE R5 TRUE NEGATIVE
         SLR   R4,R4              CLEAR R4 TO ZERO
E1S      CL    R5,D1              R5 LESS THAN X'D5000000' ?
         BC    11,E2S             NO
         SLDL  R4,8               SHIFT FIRST 8 BITS OF R5 INTO R4 AS INDEX
         IC    R4,0(R4,R13)       OBTAIN CONSTANT FROM TABLE
         STC   R4,PWRD+1          STORE IN SECOND BYTE OF PWRD
         SRL   R5,8               SHIFT REMAINING 24 BITS RIGHT THEN OR ON
         ALR   R5,R10             EXPONENT TO MAKE .(24 BITS)/16
```

```
         ST    R5,0(R7,R6)     STORE IN CALLER'S ARRAY                     LLRA5720
         LE    FR0,PWRD        LOAD CHARACTERISTIC TO FLOATING POINT       LLRA5730
         AE    FR0,0(R7,R6)    REGISTER 0 AND ADD FRACTION                 LLRA5740
         STE   FR0,0(R7,R6)    STORE EXPONENTIAL DEVIATE IN ARRAY          LLRA5750
         LR    R5,R0           COPY BACK TO R5 FOR NEXT GO AROUND          LLRA5760
         BR    R12             GO TO BXLE AND CONTINUE                     LLRA5770
E2S      CL    R5,D2           R5 LESS THAN X'F1700000' ?                  LLRA5780
         BC    11,E3S          NO                                          LLRA5790
         SLDL  R4,12           SHIFT FIRST 12 BITS OF R5 INTO R4           LLRA5800
         SL    R4,D1M          SUBTRACT OFF                                LLRA5810
         IC    R4,0(R4,R13)    OBTAIN CONSTANT FROM TABLE                  LLRA5820
         STC   R4,PWRD+1       STORE AS SECOND BYTE OF PWRD                LLRA5830
         SRL   R5,8            SHIFT REMAINING 20 BITS RIGHT THEN OR ON    LLRA5840
         ALR   R5,R10          EXPONENT TO MAKE .(20 BITS)/16              LLRA5850
         ST    R5,0(R7,R6)     STORE IN CALLER'S ARRAY                     LLRA5860
         LE    FR0,PWRD        LOAD CHARACTERISTIC TO FLOATING POINT       LLRA5870
         AE    FR0,0(R7,R6)    REGISTER 0 AND ADD FRACTION                 LLRA5880
         STE   FR0,0(R7,R6)    STORE EXPONENTIAL DEVIATE IN ARRAY          LLRA5890
         LR    R5,R0           COPY BACK TO R5 FOR NEXT GO AROUND          LLRA5900
         BR    R12             GO TO BXLE AND CONTINUE                     LLRA5910
E3S      ST    R5,EWRD         STORE R5 IN ARGUMENT LIST                   LLRA5920
         ST    R0,XWRD         PASS STARTING VALUE                         LLRA5930
         LA    R13,SA2         LOAD LOW SAVE AREA POINTER                  LLRA5940
         LA    R1,ELIST        ARGUMENT LIST FOR CALL TO REXPTH            LLRA5950
         LR    R8,R15          COPY BASE FOR BALR LINKAGE                  LLRA5960
         L     R15,AREXP       ADDRESS OF FUNCTION SUBROUTINE REXPTH       LLRA5970
         BALR  R14,R15         BRANCH TO REXPTH                            LLRA5980
         LR    R15,R8          RESTORE BASE REGISTER                       LLRA5990
         STE   FR0,0(R7,R6)    STORE EXPONENTIAL DEVIATE IN ARRAY          LLRA6000
         L     R5,XWRD         NEW STARTING VALUE                          LLRA6010
         LA    R13,BTBLE       RESTORE R13 TO TABLE OF CONTENTS            LLRA6020
         LA    R8,TABLE        RESTORE R8 TO ADDRESS OF SHUFFLING TABLE    LLRA6030
         L     R1,MASK         RESTORE R1 TO INDEX MASK                    LLRA6040
N8       BXLE  R7,R2,L8        LOOP AROUND AGAIN                           LLRA6050
         L     R13,SA2+4       RESTORE HIGH SAVE AREA POINTER              LLRA6060
         L     R1,24(,R13)     GET ARGUMENT LIST POINTER AGAIN             LLRA6070
         L     R4,0(,R1)       GET STARTING VALUE ADDRESS AGAIN            LLRA6080
         ST    R5,0(,R4)       STORE AS STARTING VALUE FOR NEXT CALL       LLRA6090
         LM    R14,R12,12(R13) RESTORE THE REGISTERS                       LLRA6100
         BCR   15,R14          RETURN                                      LLRA6110
```

```
***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

*                                                                      LLRA6130
*      CONSTANTS AND STORAGE                                           LLRA6140
*                                                                      LLRA6150
SA       DS    18F                                                     LLRA6160
SA2      DS    18F                                                     LLRA6170
PICA     DS    F                                                       LLRA6180
PM2      DC    F'2147483645'                                           LLRA6190
A75      DC    F'16807'                                                LLRA6200
         DC    X'40000001'                                             LLRA6210
         DC    X'40100000'                                             LLRA6220
A75N     DC    F'16807'                                                LLRA6230
         DC    X'3F000000'                                             LLRA6240
         DC    X'00000040'                                             LLRA6250
MASK     DC    X'0000007F'                                             LLRA6260
PWRD     DC    X'41AA0000'                                             LLRA6270
NWRD     DC    X'C1AA0000'                                             LLRA6280
C1       DC    X'68000000'                                             LLRA6290
C2       DC    X'D0000000'                                             LLRA6300
C3       DC    X'E2F00000'                                             LLRA6310
C4       DC    X'F5E00000'                                             LLRA6320
C1M      DC    X'00000068'                                             LLRA6330
C2M      DC    X'00000CE8'                                             LLRA6340
C3M      DC    X'00000E17'                                             LLRA6350
D1       DC    X'D5000000'                                             LLRA6360
D2       DC    X'F1700000'                                             LLRA6370
D1M      DC    X'00000CFF'                                             LLRA6380
FWRD     DS    F                                                       LLRA6390
EWRD     DS    F                                                       LLRA6400
XWRD     DS    F                                                       LLRA6410
FLIST    DC    AL4(FWRD)                                               LLRA6420
         DC    X'80'                                                   LLRA6430
         DC    AL3(XWRD)                                               LLRA6440
ELIST    DC    AL4(EWRD)                                               LLRA6450
         DC    X'80'                                                   LLRA6460
         DC    AL3(XWRD)                                               LLRA6470
AINT     DC    V(INT)                                                  LLRA6480
ASEXPO   DC    V(SEXPON)                                               LLRA6490
ARNOR    DC    A(RNORTH)                                               LLRA6500
AREXP    CC    A(REXPTH)                                               LLRA6510
TABLE    DC    X'347A50E5',X'326C0AF7',X'0DE685A8',X'769727F9'         LLRA6520
         DC    X'1BBA2C16',X'10BABA9C',X'11942E23',X'39CC478E'         LLRA6530
         DC    X'2F9F9D24',X'16A4845F',X'07BFFABA',X'74C3EE15'         LLRA6540
         DC    X'4006DD4F',X'28950373',X'2CC7A9DB',X'480E58B7'         LLRA6550
         DC    X'29798C8B',X'635C8633',X'64499EBC',X'7649685F'         LLRA6560
         DC    X'1123F9C8',X'43D23E61',X'5F3F4B08',X'7826F090'         LLRA6570
         DC    X'5832018E',X'5786A990',X'62C6EC83',X'13734F9C'         LLRA6580
         DC    X'2896D594',X'567C9832',X'4A977ADC',X'0E5EC953'         LLRA6590
         DC    X'6C300744',X'6BB79409',X'2DF319E0',X'2B3C1360'         LLRA6600
         DC    X'6C9B357A',X'1C0C8FF0',X'3800A4BF',X'40A95CE4'         LLRA6610
```

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
         DC    X'76839B19',X'0DB17B2F',X'6918516B',X'7B8F0F60'          LLRA6620
         DC    X'442A0D52',X'20AB7919',X'2070A25E',X'7611BBA9'          LLRA6630
         DC    X'474849DF',X'7C8B0DEB',X'53D023CE',X'6D9FEF46'          LLRA6640
         DC    X'7D91C530',X'0B478BB4',X'45C26E9F',X'4DBF9AC7'          LLRA6650
         DC    X'7366FDCB',X'305629D9',X'2A1C7AF8',X'1C6DCB3B'          LLRA6660
         DC    X'4DE86B25',X'312260FF',X'58A65D3C',X'5EDBA816'          LLRA6670
         DC    X'5066686D',X'09440578',X'0FD8671D',X'55D53EE2'          LLRA6680
         DC    X'2ED81E2F',X'44AB03B8',X'66EFEB57',X'52C5A3FB'          LLRA6690
         DC    X'6D8DDF2B',X'685F1A04',X'061B4869',X'0CAFD12A'          LLRA6700
         DC    X'3E60285D',X'3A3282F8',X'6B1DD5ED',X'18891EB4'          LLRA6710
         DC    X'0F752CD0',X'1D54304C',X'44B3AA7D',X'40D52733'          LLRA6720
         DC    X'148D6223',X'58142A4B',X'303AAF44',X'6949EC28'          LLRA6730
         DC    X'6494F236',X'403AC1E7',X'719CF6DA',X'13983F75'          LLRA6740
         DC    X'52464E2F',X'14402C96',X'76D4AB59',X'3ED2E3CF'          LLRA6750
         DC    X'779C3D29',X'607A9BFD',X'4D8A3824',X'0ACD66BD'          LLRA6760
         DC    X'67625F57',X'360BA8E2',X'249A5E43',X'412497D9'          LLRA6770
         DC    X'465A79D1',X'2264B308',X'53D28497',X'26A0D19D'          LLRA6780
         DC    X'3A0B1DB4',X'34AF002B',X'5D1FAE1C',X'65AA201A'          LLRA6790
         DC    X'521C9024',X'6DFAC836',X'0D25924E',X'2DD327FF'          LLRA6800
         DC    X'2213B94F',X'000D955C',X'0CB02D6F',X'6CC25FB7'          LLRA6810
         DC    X'5DD061F1',X'1AD4BB0A',X'10FA6C98',X'7AF65CAE'          LLRA6820
         DC    X'02D9E6D8',X'694B7943',X'5E31F4A9',X'0AE725FE'          LLRA6830
ATBLE    DC    X'00010202',X'03030303',X'04040404',X'04090A0A'          LLRA6840
         DC    X'0A0A0A0E',X'0E0E1217',X'00000000',X'00C1C101'          LLRA6850
         DC    X'01010202',X'02020303',X'04050505',X'05C5C606'          LLRA6860
         DC    X'0606C607',X'07070707',X'08080808',X'08C90909'          LLRA6870
         DC    X'090B0B0B',X'0B0C0C0C',X'0C0D0D0D',X'0DCECF0F'          LLRA6880
         DC    X'0F101010',X'11111112',X'12131314',X'14151516'          LLRA6890
         DC    X'16171819',X'1A1B1C1D',X'05050505',X'05C50505'          LLRA6900
         DC    X'05050606',X'06060606',X'06070707',X'07C7C808'          LLRA6910
         DC    X'0B0B0B0B',X'0B0B0B0B',X'0B0C0C0C',X'0C0CCD0F'          LLRA6920
         DC    X'0F0F0F0F',X'0F0F0F0F',X'0F101010',X'1C1C1C10'          LLRA6930
         DC    X'11111113',X'13131313',X'13131313',X'13131314'          LLRA6940
         DC    X'14141414',X'14141414',X'15151515',X'15161618'          LLRA6950
         DC    X'18181818',X'18181818',X'18181818',X'19191919'          LLRA6960
         DC    X'19191919',X'19191A1A',X'1A1A1A1A',X'1A1B1B1B'          LLRA6970
         DC    X'1B1B1C1C',X'1E1E1E1E',X'1E1E1E1E',X'1E1E1E1E'          LLRA6980
         DC    X'1E1E1E1F',X'1F1F1F1F',X'1F1F1F1F',X'1F1F1F1F'          LLRA6990
         DC    X'20202020',X'20202020',X'20202020',X'21212121'          LLRA7000
         DC    X'21212121',X'21212222',X'22222222',X'22222223'          LLRA7010
         DC    X'23232323',X'23232324',X'24242424',X'24242525'          LLRA7020
         DC    X'25252525',X'26262626',X'26272727',X'27282828'          LLRA7030
         DC    X'2929292A',X'2A2B2B'                                    LLRA7040
BTBLE    DC    X'00000000',X'00000000',X'00000000',X'000000C1'          LLRA7050
         DC    X'01010101',X'01010101',X'01010101',X'02020202'          LLRA7060
         DC    X'02020202',X'02030303',X'03030606',X'06C6C608'          LLRA7070
         DC    X'08080808',X'0808080A',X'0A0A0A0A',X'0A0A0A0C'          LLRA7080
         DC    X'0C0C0C0C',X'0C0E0E11',X'11151515',X'15192020'          LLRA7090
         DC    X'28010202',X'02020303',X'03030303',X'03C4C404'          LLRA7100
```

***** NAVAL POSTGRADUATE SCHOOL RANDOM NUMBER GENERATOR : LLRANDOM *****

```
         DC    X'04040404',X'04040404',X'05050505',X'05C5C505'        LLRA7110
         DC    X'05050606',X'06060607',X'07070707',X'07C7C707'        LLRA7120
         DC    X'08090909',X'09090909',X'090B0B0B',X'0B0B0B0B'        LLRA7130
         DC    X'0C0D0D0D',X'0D0D0D0E',X'0E0E0E0F',X'0FCFCF0F'        LLRA7140
         DC    X'10101010',X'10111111',X'12121212',X'13131313'        LLRA7150
         DC    X'14141414',X'16161617',X'17171818',X'1819191A'        LLRA7160
         DC    X'1A1B1B1C',X'1C1D1D1E',X'1E1F1F21',X'22232425'        LLRA7170
         DC    X'26272829',X'2A050505',X'05050707',X'090B0D0D'        LLRA7180
         DC    X'0D0D0F0F',X'0F0F0F0F',X'0F0F0F10',X'10101212'        LLRA7190
         DC    X'12121212',X'12121212',X'13131313',X'13161616'        LLRA7200
         DC    X'16161616',X'16161717',X'17171717',X'18181A1A'        LLRA7210
         DC    X'1A1A1A1A',X'1A1A1A1A',X'1A1A1A1B',X'1B1B1B1B'        LLRA7220
         DC    X'1B1B1B1B',X'1B1C1C1C',X'1C1C1C1C',X'1D1D1D1D'        LLRA7230
         DC    X'1D1E1E21',X'21212121',X'21212121',X'21212121'        LLRA7240
         DC    X'22222222',X'22222222',X'22222223',X'23232323'        LLRA7250
         DC    X'23232323',X'24242424',X'24242424',X'25252525'        LLRA7260
         DC    X'25252626',X'26262627',X'27272728',X'28292C2C'        LLRA7270
         DC    X'2C2C2C2C',X'2C2C2C2C',X'2C2C2C2C',X'2C2D2D2D'        LLRA7280
         DC    X'2D2D2D2D',X'2D2D2D2D',X'2D2D2D2E',X'2E2E2E2E'        LLRA7290
         DC    X'2E2E2E2E',X'2E2E2E2E',X'2F2F2F2F',X'2F2F2F2F'        LLRA7300
         DC    X'2F2F2F2F',X'30303030',X'30303030',X'30303031'        LLRA7310
         DC    X'31313131',X'31313131',X'31313232',X'32323232'        LLRA7320
         DC    X'32323232',X'33333333',X'33333333',X'33343434'        LLRA7330
         DC    X'34343434',X'34343535',X'35353535',X'35353636'        LLRA7340
         DC    X'36363636',X'36363737',X'37373737',X'37383838'        LLRA7350
         DC    X'38383838',X'39393939',X'39393A3A',X'3A3A3A3A'        LLRA7360
         DC    X'3B3B3B3B',X'3B3B3C3C',X'3C3C3C3D',X'3D3D3D3D'        LLRA7370
         DC    X'3E3E3E3E',X'3F3F3F3F'                                LLRA7380
R0       EQU   0                                                      LLRA7390
R1       EQU   1                                                      LLRA7400
R2       EQU   2                                                      LLRA7410
R3       EQU   3                                                      LLRA7420
R4       EQU   4                                                      LLRA7430
R5       EQU   5                                                      LLRA7440
R6       EQU   6                                                      LLRA7450
R7       EQU   7                                                      LLRA7460
R8       EQU   8                                                      LLRA7470
R9       EQU   9                                                      LLRA7480
R10      EQU   10                                                     LLRA7490
R11      EQU   11                                                     LLRA7500
R12      EQU   12                                                     LLRA7510
R13      EQU   13                                                     LLRA7520
R14      EQU   14                                                     LLRA7530
R15      EQU   15                                                     LLRA7540
FR0      EQU   0                                                      LLRA7550
FR2      EQU   2                                                      LLRA7560
         END                                                          LLRA7570
```